

Java Script

What is JavaScript ?

- Designed to add interactivity to HTML pages
- Scripting language
- JavaScript is lines of executable computer code
- JavaScript is usually embedded directly in HTML pages
- JavaScript is an interpreted language
- JavaScript is supported by all major browsers

Advantages of java Script

- **Less server interaction:** You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server
- **Immediate feedback to the visitors:** They don't have to wait for a page reload to see if they have forgotten to enter something
- **Increased interactivity:** You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard

What Java Script can do

- JavaScript gives HTML designers a programming tool
- JavaScript supports events
- JavaScript can manipulate HTML elements
- JavaScript can be used to validate data
- JavaScript can be used to detect the visitor's browser
- JavaScript has built-in objects like Date, String, Array, Math

Where to place Java Script

- Java Script can be put in the <body> and in the <head> sections of an HTML page
- Java Script in <BODY> is executed when the page loads.

```
<html>
<body>
<p id="demo">A Paragraph</p>
<script>
document.getElementById("demo").innerHTML="Hello World";
</script>
</body>
</html>
```

Where to place Java Script (contd)

- If you want to have a script run on some event, such as when a user clicks somewhere, then you will place that script in the head as a **function**
- Functions are normally used in combination with **events**

```
<head>
  <script>
    function myFunction()
    {
      document.getElementById("demo").innerHTML="Hello World";
    }
  </script>
</head>
```

Using external Script

```
document.write("Hello There")
```

```
<html>
```

```
<head>
```

```
</head>
```


```
<body>
```

```
<script src="first.js">
```

```
</script>
```

```
</body>
```

```
</html>
```



Store this line
in first.js

Java Script Syntax

- Java Script syntax is loosely based on Java syntax
- Java Script is case sensitive
- Comments are same as in Java
- White space between words and tabs are ignored
- Semicolon is optional at the end of the statement

Data Types

- JavaScript allows you to work with three primitive data types:

Numbers eg. 123, 120.50 etc.

Strings of text e.g. "This text string" etc.

Boolean e.g. true or false

- JavaScript also defines two trivial data types, *null* and *undefined*

Variables

- Variable names are case sensitive
- They must begin with a letter or the underscore character
- JavaScript is *untyped* language. Variable can hold a value of any data type
- create a variable with the var statement:

```
var strname = "Hello"
```

```
var abc, xyz;
```

```
var num = 30
```

- variable can also be created without the var statement:

```
strname = "Hello"
```

```
num = 30
```

Scope of Variables

- Variable within a function can only be accessed within that function.
- When you exit the function, the variable is destroyed.
- These variables are called local variables
- Variable declared outside a function can be accessed by all the functions in the page
- These variables are Global variables
- The lifetime of these variables starts when they are declared, and ends when the page is closed.

Operators

Arithmetic Operators + - * / % ++ --

Assignment Operators = += -= *= /= %=

Relational Operators == != > < >= <=

Logical Operators && || !

String Operator + (concatenation)

Functions

- A function contains some code that will be executed by an event or a call to that function.
- A function is a set of statements.
- We can reuse functions within the same script, or in other documents.
- We define functions at the beginning of a file (in the head section), and call them later in the document
- Functions that will return result must use the "return" statement.

Defining Functions

With Arguments

```
function myfunction(arg1,arg2,..)  
{  
  some statements  
}
```

Without Arguments:

```
function myfunction()  
{  
  some statements  
}
```

Functions Example

```
function total(a,b)
{
  result=a+b
  return result
}
```

Calling the above function

```
sum = total(23, 45)
```

Function literal

- A function literal is an expression that defines an unnamed function
- Format :

```
var variablename = function(Argument List) { Function Body };
```

- Example :

```
var func = function(x,y){ return x*y };
```

```
func(10,20); // This will produce 200
```


Conditional Statements

In JavaScript we have the following conditional statements:

if statement - use this statement to execute some code only if a specified condition is true

if...else statement - use this statement to execute some code if the condition is true and another code if the condition is false

switch statement - use this statement to select one of many blocks of code to be executed

Same syntax as Java language

Loop Statements

In JavaScript we have the following loop statements which are similar to Java loops

- while loop
- for loop
- break statement
- continue statement
- Labeled loops

In addition we have for in loop

- for (variable in object) - variable takes one property at a time

Arrays

- An array can be defined in three ways

```
var cars=new Array(); // regular array (add an optional integer  
cars[0]="Saab";        // argument to control array's size)  
cars[1]="Volvo";  
cars[2]="BMW";
```

```
var cars=new Array("Saab","Volvo","BMW"); // condensed array
```

```
var cars=["Saab","Volvo","BMW"]; // literal array
```

Arrays

Methods/ Properties	Description
length	It reflects the number of elements in an array
pop()	It removes the last element from an array and returns that element.
push()	It adds one or more elements to the end of an array and returns the new length of the array.
shift()	removes the first array element and moves all other elements to a lower index
unshift()	adds a new element at the beginning and moves other elements to next index
join()	It joins all elements of an array into a string.
reverse()	It reverses the order of the elements of an array.
sort()	Sorts array alphabetically

refer [w3schools.com](https://www.w3schools.com) for complete reference

Objects

- JavaScript is an Object Oriented Programming (OOP) language
- Objects are composed of attributes
- If an attribute contains a function, it is considered to be a method of the object otherwise, the attribute is considered a property
- Object properties can be any of the three primitive data types, or any of the abstract data types, such as another object
- Object properties are usually variables that are used internally in the object's methods, but can also be globally visible variables
- All user-defined objects and built-in objects are descendants of an object called Object

Creating Objects

- Creating direct instance of an object :

```
person=new Object();  
person.firstname="John";  
person.lastname="Doe";  
person.age=50;  
person.eyecolor="blue";
```

- Alternate way of creating an object :

```
person={firstname:"John",lastname:"Doe",age:50,eyecolor:"blue"};
```

Objects as Associative Arrays

- In addition to the dot address method of identification such as ***object.property***, object data can also be represented in array format such as ***object[property]***
- One advantage of the array representation is that the array index is a string and can thus be manipulated dynamically

```
person={firstname:"John",lastname:"Doe",age:50,eyecolor:"blue"};
```

```
var name = person.firstname;    // dot notation
```

```
var name= person["firstname"];  // array notation
```

Built-in objects

Built-in functions

Functions	Description
isNaN()	Returns true , if the object is Not a Number. Returns false , if the object is a number.
parseFloat (string)	If the string begins with a number, the function reads through the string until it finds the end of the number; cuts off the remainder of the string and returns the result. If the string does not begin with a number, the function returns NaN.
parseInt (string)	If the string begins with an integer, the function reads through the string until it finds the end of the integer, cuts off the remainder of the string and returns the result. If the string does not begin with an integer, the function returns NaN (Not a Number).

Math

- Math is static object used to perform math operations

Methods	Description
abs()	Returns the absolute value of a number.
acos()	Returns the arccosine (in radians) of a number.
ceil()	Returns the smallest integer greater than or equal to a number.
cos()	Returns cosine of a number.
floor()	Returns the largest integer less than or equal to a number.
log()	Returns the natural logarithm (base E) of a number.
max()	Returns the largest of zero or more numbers.
min()	Returns the smallest of zero or more numbers.
pow()	Returns base to the exponent power, that is base exponent.

Date

- Date is data type to perform date operations

Methods	Description
<code>new Date()</code>	Creates Date object for today
<code>new Date(year,month,date,.....)</code>	Creates Date object for given date
<code>getDate()</code>	Returns the day of the month.
<code>getDay()</code>	Returns the day of the week.
<code>getFullYear()</code>	Returns the year.
<code>getHours()</code>	Returns the hour.
<code>getMinutes()</code>	Returns the minutes.
<code>getSeconds()</code>	Returns the seconds.
<code>getMonth()</code>	Returns the month.

refer [w3schools.com](https://www.w3schools.com) for complete reference

String

- String is data type to perform String operations

Methods	Description
charAt()	It returns the character at the specified index.
indexOf()	It returns the index within the calling String object.
match()	It is used to match a regular expression against a string.
replace()	It is used to replace the matched substring with a new substring.
search()	It executes the search for a match between a regular expression.
slice()	It extracts a section of a string and returns a new string.
split()	It splits a string object into an array of strings by separating the string into the substrings.
toLowerCase()	It returns the calling string value converted lower case.
toUpperCase()	Returns the calling string value converted to uppercase.

refer [w3schools.com](https://www.w3schools.com) for complete reference

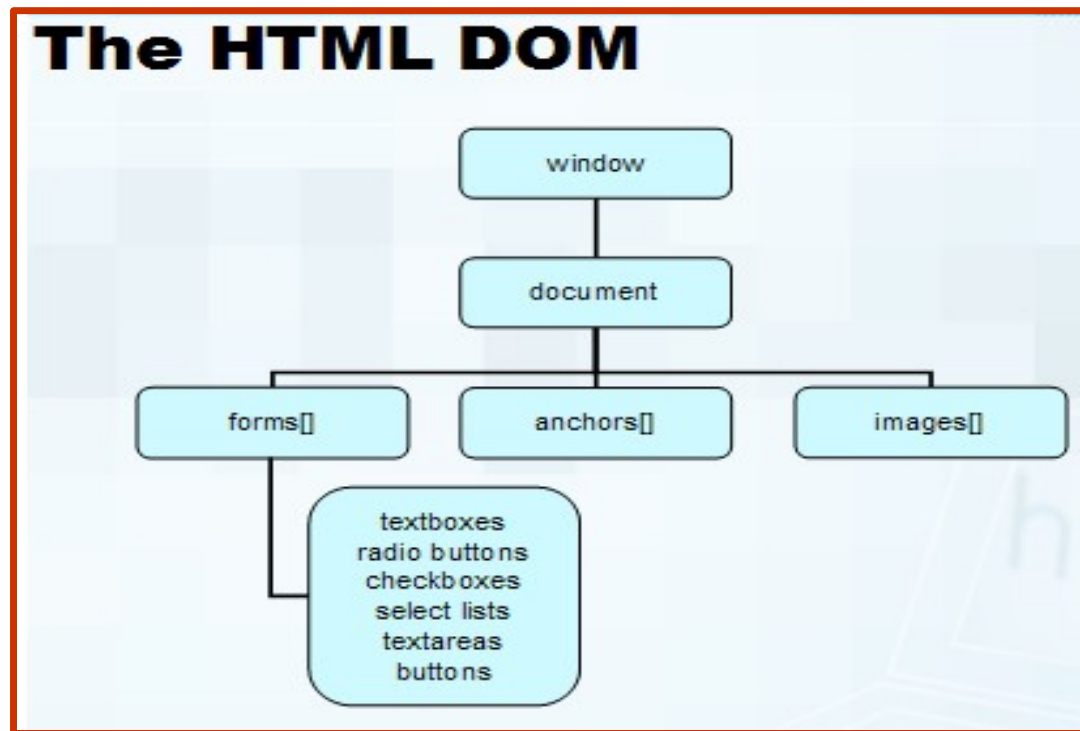
HTML DOM

What is DOM ?

- *The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document*
- The DOM is separated into 3 different parts / levels:
 - Core DOM - standard model for any structured document
 - XML DOM - standard model for XML documents
 - HTML DOM - standard model for HTML documents
- The DOM defines the **objects and properties** of all document elements, and the **methods** (interface) to access them
- JavaScript can be used to create and destroy these objects, to invoke their methods, and to manipulate their properties

HTML DOM

- A standard object model for HTML
- The HTML DOM defines the objects and properties of all HTML elements, and the methods (interface) to access them.
- The HTML DOM is a standard for how to get, change, add, or delete HTML elements.



Element nodes

- Element nodes can be accessed by their tag name, id, or position within the HTML document hierarchy

getElementsByTagName()

This method of an element node retrieves all descendant elements of the specified tag name and stores them in a NodeList, which is exposed in JavaScript as an array

getElementById()

This method of the document node returns a reference to the node with the specified id

DOM events

- Every element on a web page has certain events which can trigger JavaScript functions
- We define the events in the HTML elements.
- Examples of events:
 - A mouse click (onclick)
 - A web page or an image loading (onload)
 - Mousing over a hot spot on the web page (onmouseover)
 - Selecting an input box in an HTML form (onselect)
 - Submitting an HTML form (onsubmit)
 - element gets focus (onfocus)
 - element loses focus (onblur)

Attaching events

- It is possible to attach all sorts of events, such as clicks, mouseovers, mouseouts, etc., to elements on the page
- The simplest method for adding events is to assign an event handler property to a node

- The syntax is shown below:

`node.onevent = DoSomething;`

- Example :

E-mail: `<input type="text" id="email" onchange="checkEmail()" />`